**I. Introduction**

Portable Instruments Company (PICo) has offered a large contract to the winning design of a specific SRAM. Of the two options, we chose to win “Design Application 1 – Low power SRAM (Mem1)” because designing a component for low power was foreign to us. While we designed an adder for high speed in a previous class, we desire to challenge ourselves with a new critical metric. The company asked for a 1Mb SRAM with 32-bit words for use in a microsensor node. We believe that we have made significant progress towards completing this project, and expect to finish by the December 1 deadline. This design review is meant to ensure that we are on the right track to meet PICo’s specifications, and that our design choices are well founded. Feedback from PICo will be instrumental in finalizing the project.

In this paper we revisit our old timeline and offer a new, more detailed one in section II. Section III presents a complete block diagram of the SRAM. The status of each component of the SRAM is also given. Because the entire SRAM is too large to efficiently simulate, a model was created to approximate the power and delay tradeoffs and is given in section IV. Although the schematics have not been fully completed, we have already begun creating the layouts of some circuit elements. Section V has these layouts. Section VI gives the results of our model simulations. This section is followed by a discussion of the progress so far and the remaining challenges. References are given in section VIII.

**II. Timeline**

Figure 1 shows our old timeline with annotations. We are making progress despite unforeseen setbacks. Figure 2 presents an updated timeline for the remainder of the project and enumerates tasks for each project member.



Figure 1. Old timeline with annotations showing how the project has unfolded so far

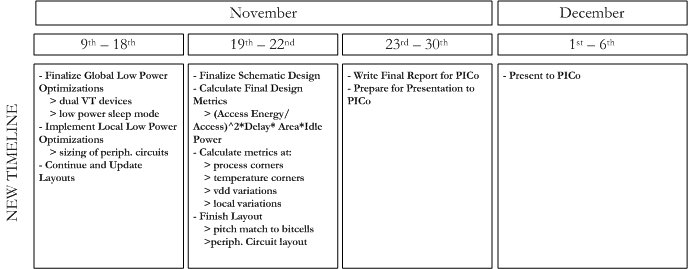


Figure 2. Updated timeline with tasks needed to complete design.

Task Breakdown:

* Special Features:
  + Dual VT – Kevin
  + Error Correction – Roger
  + Sleep mode – Stevo
* Sizing of periphery circuits – Kevin, Stevo
* Synchronizing SRAM – Kevin
* Full schematics – Mostly Roger, Stevo
* Full layout – Mostly John, everyone
* Monte Carlo variation tests - John
* Final metric calculation – everyone
* Final report and presentation - everyone

**III. Block Diagram**

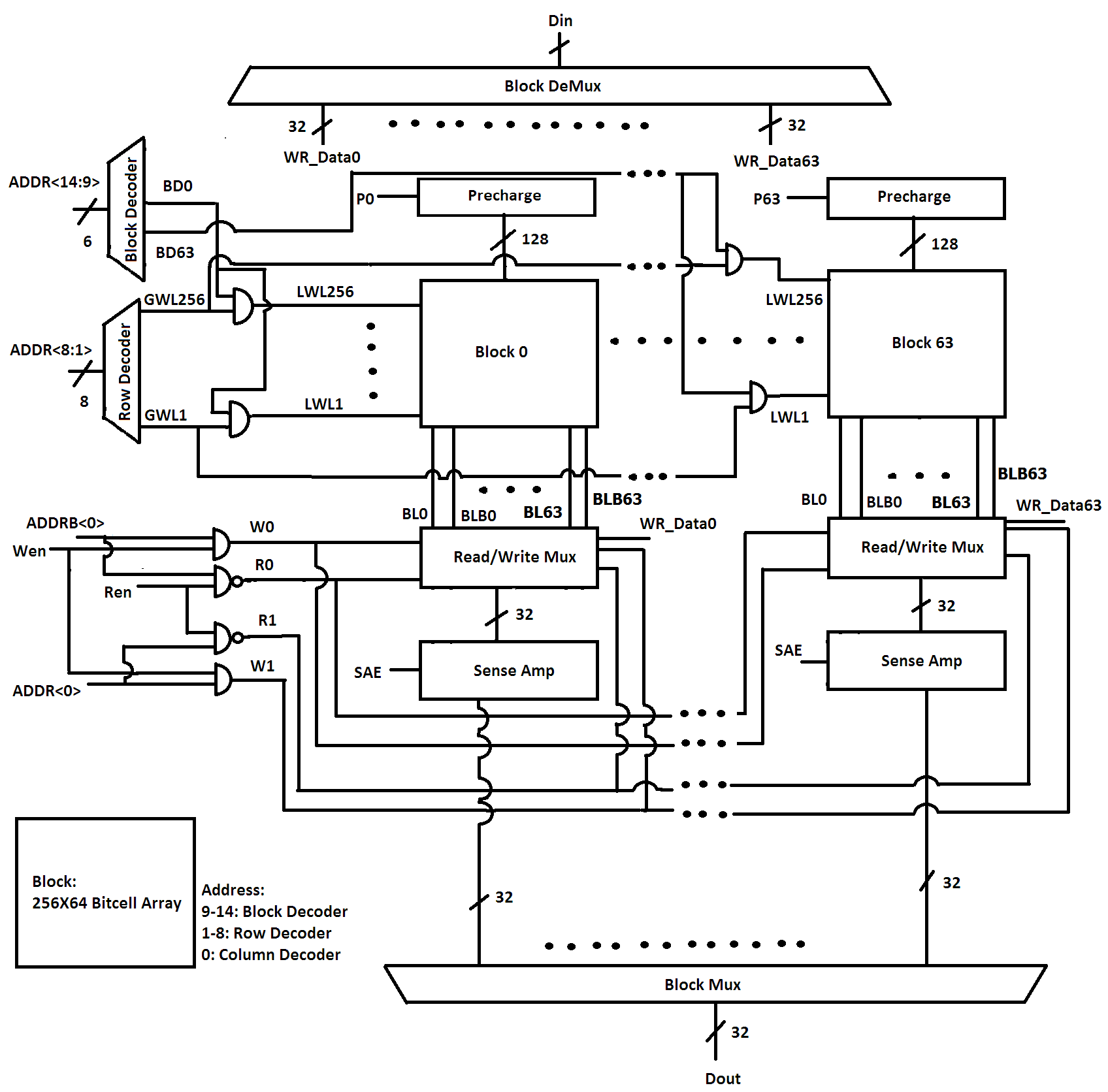


Figure 3. Block diagram of the full SRAM

Table 1. Component status list



**IV. Simulation Model**

To find the optimal number of blocks and optimal size of each block, we created and simulated a model of our SRAM. The model uses parameters and multipliers to accurately size itself for a given number of rows (*r*), columns (*c*), and blocks (*b*). In this section, each part of the model is described and justified.

1. Blocks

The block used for reading and writing simulations is modeled by four minimum-sized bitcells and four load bitcells. The minimum-sized bitcells allow us to read and write to a normal bitcell with different column and row address values. The load bitcells provide capacitance to the word lines and bitlines. They have multipliers which are dependent on *c* and *r* to vary the load for a given block size. Unused word lines are grounded, and unused bitlines are connected to VDD. In addition to delay modeling, this model approximates the power consumed through leakage of the bitlines. The two unconnected bitlines still leak into the bitcells, and this current comes from VDD. Figure 4 below shows the block model, which was suggested by Professor Calhoun. Another block model was required to estimate the power consumed by the rest of the blocks, which are always off during our simulations. This model consists of leaking bitcells with multipliers proportional to *b* and *c* or *r*, as seen in Figure 5.



Figure 4. Block model for the bitcells under simulation



Figure 5. Block model used to find energy drawn from leaky, unused bitcells

1. Precharge

Our precharge circuit has two PMOSs, one for each bitline, and an equalizing PMOS coupling the two bitlines. The idea came from [1]. For the model, we have one minimum-sized precharge block for each pair of bitlines we simulate, one upsized precharge block for the remaining bitlines in the block, and one upsized precharge block for all the other blocks. The upsized precharge blocks model the power and capacitive load of all the precharge blocks. Each precharge block’s size can vary for different *c* and *b* values. To minimize precharge signal delay, we are buffering the precharge input to each block; hence we have two precharge signals for our model: P1 and P2. Figure 6 shows the precharge blocks and their respective signals.



Figure 6. Model of the precharge blocks used for loads and power estimation

1. Decoders

After deciding on a block size and number of blocks (see section VI), we added the decoders to our model. Because varying the depth of the decoders with different block sizes proved difficult to model, we left them out of the initial simulations. Since the size and number of blocks themselves were more important than the delay and power of the decoders, this design choice is justified. There are only 64 columns per block in our SRAM, so the column decoder simply selects between the two words. The column decoder used for our model can be seen in the global block diagram of section III. Figures 7 and 8 show the load and power models for the block decoder and the row decoder. This optimal type of decoder resulted from [2].



Figure 7. Model of the block decoder with predecoders (the three-input AND gates)



Figure 8. Model of the row decoder with predecoders

1. Word Lines

The row decoder outputs to the global word lines (GWL). These word lines are long, so we added an RC load to model the wire delay. The values of the resistor and capacitor are proportional to the length of the wire, *bc*.They feed into AND gates which use the output of the block decoder to select a local word line (LWL). Since the local word lines have a length proportional to *c*, we chose to add an RC load to these as well. Extra AND gates are used to model the load of the GWL signal, as seen in Figure 9. Two word line models are needed, one for each set of simulated bitcells.



Figure 9. Word line model with wire parasitics and block select gates

1. Read/Write Mux

In order to limit the number of sense and write amplifiers we need, read and write muxes are appended to the bitlines. Reference [1] provided the general idea of using PMOS pass gates for the read mux and NMOS pass gates for the write mux. This allows the bitlines to pass full VDD signals to the sense amps and the write amps to pull the bitlines all the way to ground. The write mux NMOS transistors are upsized to speed up the discharging of the bitlines. WAO and WAOB are the outputs of the write amplifier.



Figure 10. Read and write muxes separating the bitlines from the amplifiers

1. Sense Amps

The two sense amps shown in Figure 11 were simulated to determine the best choice. Sense amp 1, a latch-type sense amp, was chosen from [1] because it provides a simple yet effective design. The other sense amps from the book consume more power. Sense amp 2, obtained from [3] and also presented in the class lecture on sense amps, was another viable option. The read delay was found for each sense amp with minimum-sized transistors and a constant bitline capacitance. Data obtained from these simulations are shown in Table 2. Because sense amp 2 is faster, we chose to use it for our model. We ignored power and area differences. The power and area of the sense amps are negligible compared to those of the bitcells. The four extra transistors for sense amp 2 amount to about 32,000 extra transistors for a 1Mb SRAM with 256 blocks and 32-bit words. However, the 1Mb SRAM has over 6.25 million transistors for the bitcells alone. Simulations showed that the sense amps consume ~10-23 joules per read, while the entire SRAM uses ~10-21 joules per read. Figure 12 shows how the sense amps were implemented in our model. Two minimum-sized sense amps were needed for the simulated bitlines. Another sense amp, upsized but turned off, modeled the power drawn by the remaining sense amps in the other blocks and columns.



Figure 11. Two latching sense amplifiers tested for the SRAM

Table 2. Sense amp read delays





Figure 12. SRAM circuits used on the simulated paths and as sources of power consumption

1. Write Amps

The write amplifiers are simply large NMOS transistors. After the block demux, the input data is put through two inverters, the outputs of which drive the write amps. Since read delay is generally greater than write delay (as seen in our results section), the write amps were not used in the model for determining the ideal partitioning of the SRAM. We only simulated a read. After we decided that the best SRAM has 64 columns, only two pairs of write amps are needed per input bit per block. Figure 13 shows these two pairs. One pair is connected to the signals of interest, WAO and WAOB. The other pair is attached to a write mux which is off. This circuit emulates the loads on the critical signal path and the leakage of the off path.



Figure 13. Write amplifier models for the two simulated paths

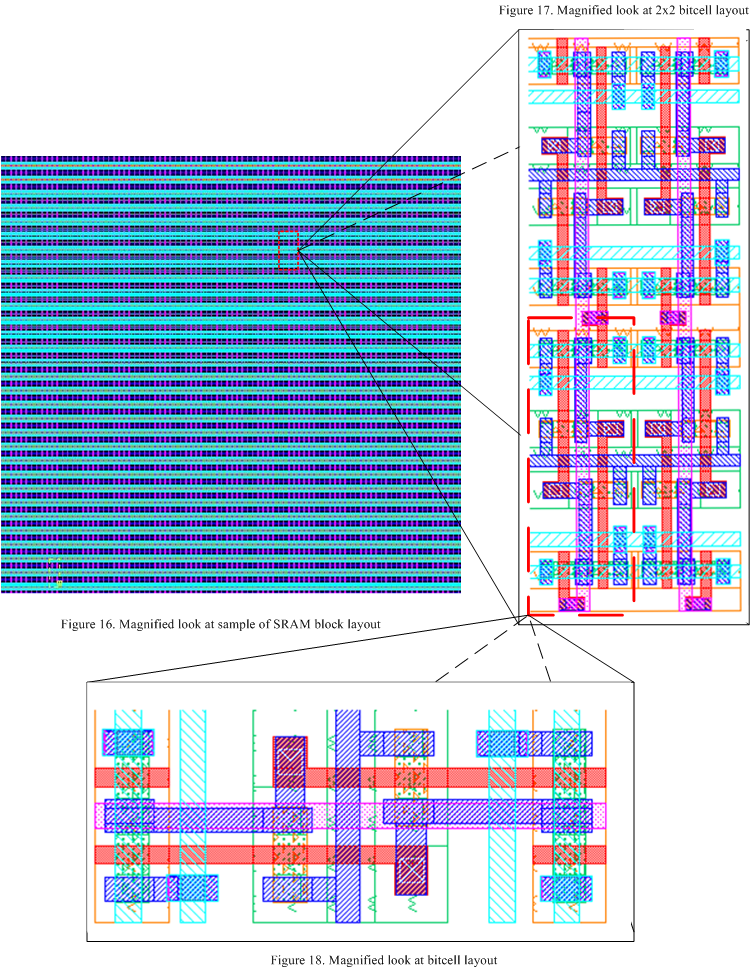
1. Block Mux/Demux

We chose to implement a simple transmission gate mux and demux for the blocks. Since only one branch will be on at a time, we can approximate this as being the only branch, with each internal node connected to one other transmission gate. Figure 14 shows how this works. Inverters buffer the signal to ensure it does not degrade too much through the transmission gates. To model leakage, we assume that the off branches of the block mux do not leak since they are connected to the output and the outputs of the sense amps, which are all low because the sense amps are off. The leakage of the block demux can be approximated by the circuit in Figure 15.

Figure 14. Block mux and demux model used on the simulated paths Figure 15. Block demux model for leakage estimating

**V. Layout**



**VI. Results**

Our model produced the energy vs. delay graphs shown in Figures 19 and 20. Figure 19 shows the results from the original model, neglecting the decoders and write circuitry. Five of the best partitions were selected and simulated with the decoders included. From Figure 20, we picked the optimal size and number of blocks, which can be seen in Table 3.

Figure 19. Energy vs. Delay graph used to narrow down the optimal block size.

Figure 20. Energy vs. Delay graph used to find the optimal size and number of blocks; the leftmost point is shown in Table 3.

Table 3. Optimal block partitioning for the 1Mb SRAM



After finding the ideal block size and number of blocks, we simulated our full SRAM model as seen in section IV. Figure 23 shows the timing diagram for model under typical-typical (TT) process conditions. For two bitcells in different columns, the initial value was read, this value was overwritten, and then the new value was read. The simulation was then run for the fast-slow (FS) condition, Figure 24, since this produced the worst-case read margin in our bitcell (see Figure 25 for read margins). Figure 25 shows this condition. Figure 26 shows the static noise margin for a hold across the process corners.

In addition these results, we obtained noise margin and Monte Carlo data for the 45nm 6T bitcell. Figures 21 and 22 shows how the static noise margin (SNM) distribution differs with process variations and varying supply voltages. As VDD decreases, the SNM falls as expected until it breaks down completely. We chose VDD = 600 mV because the SNM for a read drops off sharply when the supply is less than 600 mV.

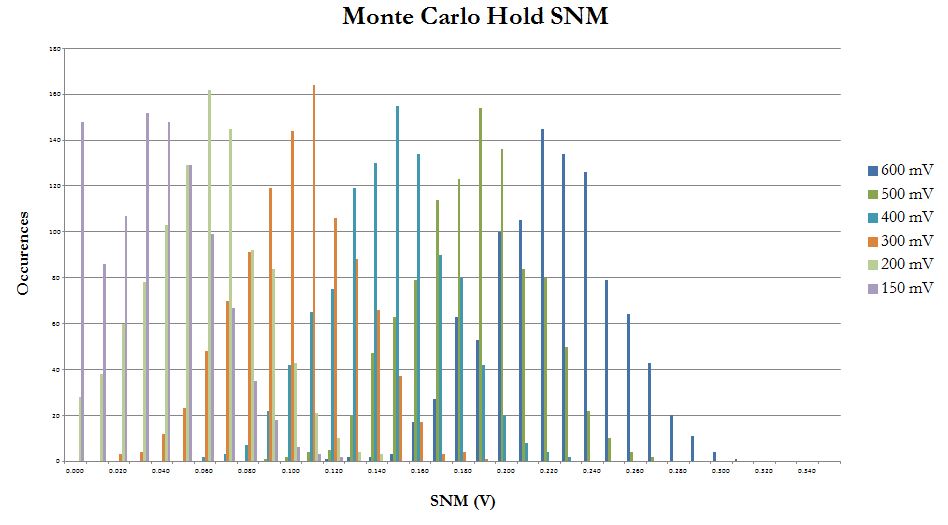


Figure 21. Monte Carlo simulation of the static noise margin for different supply voltages in Histogram form

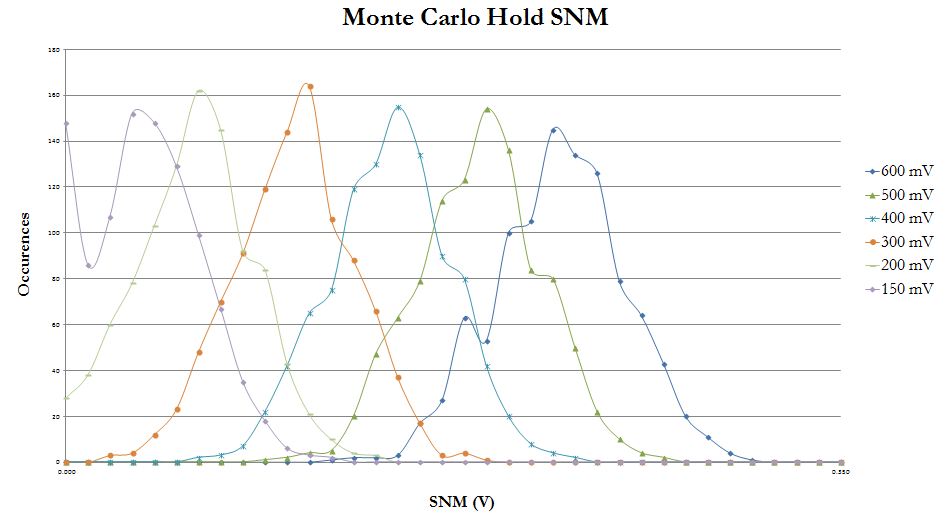


Figure 22. Monte Carlo simulation of the static noise margin for different supply voltages in linear form.

**VII. Remaining Tasks**

The remaining tasks for our project can be broadened to: finalizing periphery circuit designs, synchronizing the SRAM, implementing special features, finish creating full schematics for LVS verification, finishing layout of all components, and obtaining our final metric. Most periphery circuit design decisions have been made, but sizing of gates needs to be further explored. We also need to create a fully synchronous system to interface with the SRAM, as most of our inputs are controlled voltage sources. We need to create registers for data, and need to have all of our control signals come from the given input signals. We plan to include more special features of our SRAM including Dual VT devices for periphery circuits such as the decoder and error correction, and a low-power sleep mode. The main obstacle we face at this point is the time remaining to finish the project. If we are able to finish these tasks quickly, we will run simulations to account for local variation of our SRAM. However, we believe that it is more vital to optimize our model. The technical challenges involved with this will be recognizing the shortcomings of our model, and correcting for them as much as possible.

**VIII. References**

[1] B. Jacob, S. W. Ng, and D. T. Wang, Memory systems: Cache, DRAM, disk, Burlington, MA: Morgan Kaufmann, 2008, p. 282.

[2] B. S. Amrutur and M. A. Horowitz, “Fast low-power decoders for RAMs,” JSSC, vol. 36, no. 10, 2001.

[3] J. F. Ryan and B. H. Calhoun, “Minimizing offset for latching voltage-mode sense amplifiers for sub-threshold operation,” ISQED, 2008.